Université d'Ottawa
Faculté de génie

École de science
d'informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical
Engineering
and Computer Science

Assignment 6
CSI2120 Programming Paradigms
Winter 2017
Due on April 7th, 2017 before 11:00 pm in Virtual Campus
[5 marks in total]

Question 1. **[3 marks]**

Consider the digits $d_k, d_{k-1}, \ldots, d_1, d_0$ of a positive integer number. The squares of the digits are then $d_k^2, d_{k-1}^2, \ldots, d_1^2, d_0^2$ and we name the sum of these squares s. We can create a recursive series of the squares of the digits of positive integers. This series will be s_0, s_1, s_2, s_3, ... where s_0 is the sum of squares of the original number, s_1 the sum of the squares of the digits of s_0, s_2 the sum of the squares of the digits of s_1, and so on. For example:
$120 \rightarrow 1^2 + 2^2 = 5 \rightarrow \ 5^2 = 25 \rightarrow 2^2 + 5^2 = 29 \rightarrow \ldots$

a) Create a function `sosd` in GO that calculates the sum of square digits.

```
func sosd( num int) int
```

It has been shown that for any starting number, the series described will always reach one of the following numbers: 0,1,4,16,20,37,42,58,89,145 (OEIS A039943; Porges 1945). In the following I call these numbers stop numbers. If the series reaches the number 1 for a starting number H, then the number H is called a 'happy number'.

b) Write a function `stop` in GO that is true if the argument is one of the stop numbers in the above list.

```
func stop( num int) bool
```

c) Create a function `sosd_series` that returns a list containing all the sum of square digits calculated until (and including) a stop number is reached.

```
func sosd_series( num int) []int
```

Hint: You can use `append` to implement a "growable array" for the slice to be returned.

d) Create a function `happy?` that returns true if the function `ssod_series` ends in a 1.

```
func happy(num int) bool
```

e) Include a main function that in a loop allows the user to enter an integer and in response shows the sosd_series and prints happy or unhappy as appropriate. Example:

```
go run sosd.go
Enter a positive number (Anything else to exit): 1411
[19 82 68 100 1]
Happy
Enter a positive number (Anything else to exit): 534
[50 25 29 85 89]
Unhappy
Enter a positive number (Anything else to exit): e
```

Question 2. **[2 marks]**

Change the following program to concurrently interpolate between the start and endpoint by turning the method linear into a go routine (a function) and sending each result on a separate channel to the main program. Use a select statement for the print loop to react to the sent interpolation values and print them as they are received. Add a timeout in the select to ensure your program terminates.

Hint: In order not having to hard-code the array indices in the select statement, you may use intermediate go routines to receive an interpolation value and resend them on a common channel.

```
import (
      "fmt"
)

type Pixel struct {
      x, y float32
}

type Line struct {
      startPoint, endPoint Pixel
}

// Linear interploation
// ToDo: turn into go routine, send result on a channel
func (l *Line) linear(t float32) *Pixel {
      return &Pixel{(1.0-t)*l.startPoint.x + t*l.endPoint.x,
            (1.0-t)*l.startPoint.x + t*l.endPoint.y}
}

func main() {
      l := Line{Pixel{1.0, 3.0}, Pixel{7.0, -2.0}}
      point := make([]Pixel, 10)
```

```
	for i, t := 0, float32(0.0); i < 10; i, t = i+1, t+0.1 {
		point[i] = *l.linear(t)
	}

	// print loop
	// ToDo: add select listening to all channels opened in linear
	for i := 0; i < 10; i++ {
		fmt.Printf("(%f,%f)\n", point[i].x, point[i].y)
	}
}
```